Explanation of Retrieval Mismatches in Recommender System Dialogues

David McSherry

School of Computing and Information Engineering, University of Ulster, Coleraine BT52 1SA, Northern Ireland
dmg.mcsherry@ulster.ac.uk

Abstract. A limitation of similarity-based retrieval in recommender systems that appears to have received little attention is the absence of a formal mechanism for explaining the *retrieval mismatches* that occur when there is no product that exactly matches the requirements of the user. To address this issue, we present techniques for generating explanations of retrieval mismatches to help users understand "what went wrong" with their queries and how to construct more successful queries.

1 Introduction

An advantage of case-based reasoning (CBR) as an approach to product recommendation is that in the absence of a product that exactly matches the user's query, she can be shown the products that are most *similar* to her query (Wilke *et al*., 1998). A basic premise in the approach is that one of the recommended cases may be acceptable to the user even if it fails to satisfy all her requirements. However, recent research has highlighted some important limitations of similarity-based retrieval. For example, the most similar case may not be the one that is most acceptable to the user (Burkhard, 1998; Smyth and McClave, 2001). A related issue is that the most similar cases also tend to be very similar to each other, with the result that they may not be sufficiently representative of *compromises* that the user may be prepared to make (McSherry, 2003). As noted by Bridge and Ferguson (2002), users may also have difficulty understanding why one product is recommended while another is not recommended.

Techniques recently developed to address these limitations include:

- Retrieval algorithms that attempt to achieve a better balance between the similarity and diversity of the retrieved cases (McSherry, 2002a; Smyth and McClave, 2001), or which deliver recommendations that are inherently diverse (Bridge and Ferguson, 2002)
- A retrieval algorithm in which similarity and compromise play complementary roles, thus increasing the likelihood that one of the recommended products will be acceptable to the user (McSherry, 2003)
- Automatically-generated explanations of why a product has been recommended in terms of the compromises it involves (McSherry, 2002b; McSherry, 2003)

Another limitation of similarity-based retrieval, and one we believe to be particularly important in product recommendation, is the absence of a formal

mechanism for explaining the retrieval *mismatches* that occur when there is no exact match for the user's query. In contrast to approaches that use exact matching and rely on *constraint relaxation* techniques to recover from retrieval *failures* (e.g., Bridge, 2002), retrieval mismatches may not even be acknowledged in similarity-based retrieval. Feedback provided, if any, tends to be limited to simple techniques such as highlighting the attributes in a recommended case whose values differ from the preferred values specified in the user's query.

Recent work in *mixed-initiative* CBR has highlighted the importance of dialogue features inspired by human problem-solving behaviours, such as explanatory feedback provided at the initiative of the system or at the request of the user (Aha *et al*., 2001; Bridge, 2002; Göker and Thompson, 2000; McSherry, 2001; McSherry, 2002c; Ricci *et al*., 2002). In a troubleshooting dialogue with a human expert, for example, it would be unusual not to receive some feedback, such as a comment that a reported symptom eliminates a certain diagnosis (McSherry, 2001). It is equally natural for a customer engaged in dialogue with a human salesperson to expect feedback, such as an explanation that a certain feature is not available in her preferred price range.

In this paper, we present techniques for generating *explanations* of retrieval mismatches to help users understand "what went wrong" with their queries and how to construct more successful queries. In Sections 2 and 3, we describe our approach and its implementation in a recommender system prototype called *ShowMe*. Related work is discussed in Section 4 and our conclusions are presented in Section 5.

2 Explanation of Retrieval Mismatches

Our approach is based on the way we believe a human salesperson would respond to a customer whose expectations she believed to be unrealistic. For example, if a customer asks for a laptop computer with a 19 inch screen made by Dell, the salesperson is likely to point out that there is no such thing as a laptop with a 19 inch screen. Implicitly, the salesperson is also telling the customer that there is no problem getting a Dell laptop, or a Dell PC (e.g., a tower system) with a 19 inch screen. While a CBR system that has no knowledge of what products are available elsewhere cannot say for certain that there is no such thing as a laptop with a 19 inch screen, it can tell the user that there is no such product in the case library.

More formally, we believe that a natural approach to explaining retrieval mismatches is to draw the user's attention to *sub-queries* of her query that are not *covered* by the case library. We say that a query is covered by the case library if there is at least one case in the case library that exactly matches the query. Typically in recommender systems, a query Q over a subset A_Q of the case attributes A is represented as a set of attribute-values pairs. We refer to $|A_Q|$ as the *length* of the query and for each $a \in A_Q$ denote by $\pi_a(Q)$ the preferred value of a that the user has specified in her query.

Definition 1. *We say that a given query Q^* is a sub-query of another query Q if $A_{Q^*} \subseteq A_Q$ and $\pi_a(Q^*) = \pi_a(Q)$ for all $a \in A_{Q^*}$.*

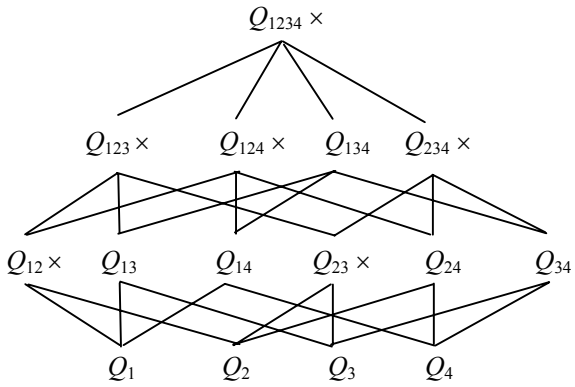

Fig. 1. Sub-queries of an example query involving four attributes. The sub-queries marked '×' are those which, in our example, are not *covered* by the case library

To illustrate our approach, Fig. 1 shows all sub-queries (apart from the empty query) of a query Q_{1234} involving four attributes a_1, a_2, a_3, and a_4. We denote by Q_{134} the sub-query involving only the attributes a_1, a_3, and a_4, by Q_{34} the sub-query involving only a_3 and a_4, and by Q_4 the sub-query involving only a_4. We use a similar notation for each of the other sub-queries of Q_{1234}. Bearing in mind that a given query is a sub-query of itself, we can see that Q_{1234} has 15 sub-queries. The four single-attribute queries of Q_{1234} are bound to be covered if, as we assume in this paper, the user's choice of an ideal value for each attribute is restricted to values that occur in the case library. However, there is no guarantee that any of the other sub-queries are covered. We will assume in this example that all sub-queries of Q_{1234} are covered except those that are marked '×' in Fig. 1.

The question is, how should the absence of a case that exactly matches the user's query be explained in this case? Our aim is to construct an explanation of the shortest possible length that will enable the user to avoid further retrieval mismatches that can be predicted from the current mismatch. For example, if our explanation mentions only that Q_{12} is not covered, the user may be tempted to try Q_{234} as her revised query. Of course, this can only lead to a further retrieval mismatch that could have been avoided with a more detailed explanation. On the other hand, if we tell the user that Q_{12} and Q_{23} are not covered, she can easily infer that Q_{123}, Q_{124}, and Q_{234} are not covered, and also that her original query Q_{1234} is not covered. The user is also entitled to infer that Q_{134} must be covered, as none of its sub-queries is mentioned in our explanation.

In general, an explanation of a retrieval mismatch (that is, the absence of an exact match for the user's query) will consist of one or more sub-queries of the user's query that are not covered by the case library, presented in non-decreasing order of query length. We are now in a position to define more formally what we mean by a *minimal* explanation of a retrieval mismatch.

Definition 2. *A minimal explanation of a retrieval mismatch for a given query Q is a sequence* $Q^1, Q^2, ..., Q^n$ *of sub-queries of Q such that:*

- *none of* $Q^1, Q^2, ..., Q^n$ *is covered by the case library*
- *for* $1 \le r \le s \le n$, $\left|A_{Q^r}\right| \le \left|A_{Q^s}\right|$
- *for* $1 \le r, s \le n$, Q^r *is not a sub-query of* Q^s
- *for any sub-query Q* of Q that is not covered by the case library, there exists* $r \le n$ *such that* Q^r *is a sub-query of Q**

Our algorithm for generating minimal explanations of retrieval mismatches, which we call *Explainer*, is outlined in Fig. 2. *SubQueries* is a list of all sub-queries, in non-decreasing order of query length, of a query for which a retrieval mismatch has occurred. *Explanation* is the (initially empty) list of sub-queries selected from *SubQueries* to explain the retrieval mismatch. For each sub-query Q^x it encounters that is not covered by the case library, *Explainer* adds Q^x to the explanation and deletes any sub-query Q^y that includes Q^x as a sub-query from the remaining list of candidate sub-queries.

```
algorithm Explainer(SubQueries, CaseLibrary)
begin
  Explanation ← ϕ
  while |SubQueries| > 0 do
  begin
    Q^x ← first(SubQueries)
    Deletions ← {Q^x}
    if Q^x is not covered by CaseLibrary
    then begin
           Explanation ← Explanation ∪ {Q^x}
           for all Q^y ∈ rest(SubQueries) do
           begin
             if Q^x is a sub-query of Q^y
             then Deletions ← Deletions ∪ {Q^y}
           end
         end
    SubQueries ← SubQueries - Deletions
  end
  return Explanation
end
```

Fig. 2. Algorithm for generating minimal explanations of retrieval mismatches

To help make our explanations of retrieval mismatches more understandable to users, they are expressed in terms of "combinations of features" in the user's query for which there is no exact match (e.g., screen size = 19, type = laptop).

3 Recommendation in ShowMe

We now present an implementation of our approach in a recommender system prototype called ShowMe. The example dialogue in Fig. 3 is based on the PC case library (McGinty and Smyth, 2002), which contains the descriptions of 120 personal computers. Attributes of the retrieved cases are listed in the order: make, processor, speed, screen size, type, memory, hard-disk size, and price. The similarity of each case to the user's query is shown in brackets. Minor editing in the interest of clarity includes the format of the queries entered by the user. Because of space restrictions, the size of the ShowMe retrieval set (the number of cases recommended on each cycle) is limited here to three.

ShowMe: Please enter your query.

User: make = Dell, processor = Intel Pentium, type = laptop, screen size = 17

ShowMe: Sorry, there is no match for the following combination of features in your query:

type = laptop, screen size = 17

However, you may wish to consider the following alternatives:

Case 33: Dell, Intel Pentium, 900, 15, laptop, 256, 10, 1345 (0.95)
Case 27: Dell, Intel Pentium, 1000, 15, laptop, 512, 30, 1429 (0.95)
Case 34: Dell, Intel Pentium, 900, 14, laptop, 256, 10, 1169 (0.93)

ShowMe: Please enter your query.

User: make = Dell, processor = Intel Pentium, screen size = 17

ShowMe: The following cases are recommended:

Case 56: Dell, Intel Pentium, 1700, 17, desktop, 512, 40, 1857 (1)
Case 55: Dell, Intel Pentium, 1900, 17, desktop, 2000, 40, 1507 (1)
Case 54: Dell, Intel Pentium, 1600, 17, desktop, 512, 20, 1417 (1)

Fig. 3. Example recommendation dialogue

The dialogue begins with ShowMe inviting the user to enter a query. Before presenting its recommendations, ShowMe searches the case library for a case that exactly matches the user's query, but is unable to find an exact match. It now examines each sub-query of the user's query to identify the cause of the retrieval mismatch and discovers that the only (proper) sub-query of her query that is not covered by the case library is:

{type = laptop, screen size = 17}

After explaining why there is no exact match for the user's query, ShowMe retrieves and displays the three products that are most similar to the user's query. In the light of the explanation provided by ShowMe, the user (who would really like a 17 inch screen) decides to revise her query by omitting her preference for a laptop. With a form-based interface, of course, it would be necessary only for the user to change a single field in her original query. On this occasion, the user's query is successful and ShowMe is able to recommend three cases (in fact there are several more) that match her requirements exactly.

The results produced in response to the user's initial query illustrate a known limitation of similarity-based retrieval, namely that the products that are most similar to the target query also tend to be very similar to each other. The three cases recommended by ShowMe match the user's query on make and type but not on screen size. While the user may be prepared to compromise on type but not on make or screen size, the retrieval set includes no alternative that offers this compromise. However, there is nothing to prevent our approach from being combined with a retrieval strategy designed to address this issue such as *diversity-conscious* retrieval (McSherry, 2002a; Smyth and McClave, 2001) or *compromise-driven* retrieval (McSherry, 2003).

4 Related Work

A growing number of conversational approaches to product recommendation rely on constraint relaxation techniques to recover from retrieval *failures* caused by the absence of a product that exactly meets the requirements of the user. Göker and Thompson's (2000) Adaptive Place Advisor is an in-car recommender system for "places to go" in which the selection of the next attribute to be constrained, or relaxed if there is no alternative that meets the requirements of the user, is based on an information gain measure. Ricci *et al.*'s (2002) Intelligent Travel Recommender (ITR) has a query relaxation component that advises the user about what to do when none of the available travel items exactly matches her requirements. For each constraint in the target query that can be relaxed without affecting the user's information goal, it tells the user how many results she will get if she relaxes that constraint. For example, the user might be told:

If you don't care about "Accept Pets" you obtain 3 results

Whereas the "best" constraint to be relaxed is suggested at the initiative of the system in the Adaptive Place Advisor, the choice of which constraint to relax is left to the user in ITR. Another system in which failure to retrieve a product that exactly matches the user's requirements results in her being asked to choose a constraint to be relaxed is Sermo, a basic prototype developed by Bridge (2002) to illustrate the potential role of dialogue grammars in conversational recommender systems.

One thing that the Adaptive Place Advisor, ITR, and Sermo have in common is their treatment of the user's requirements, or at least some of her requirements, as hard constraints. Their use of query relaxation as a means of recovery from retrieval failure is a significant improvement on traditional database approaches that leave the

user with no option but to try again with a revised query when there is no product that exactly matches her requirements (Wilke *et al*., 1998). However, a limitation of constraint relaxation is that recovery may not be possible by relaxing a single constraint. In a recommender system for cars, for example, the following query is bound to fail as Peugeot cars originate in France (or Great Britain) and a car can be described as a hatchback only if it has 3 doors or 5 doors:

{make = Peugeot, origin = Japan, body style = hatchback, doors = 4}

No matter which constraint the user chooses to relax, there will still be no exact match for her query.

In contrast, our approach to explaining retrieval mismatches works equally well even if the problem cannot be resolved by relaxing a single constraint. If we assume in the above example that the available products are adequately covered by the case library, then the user would simply be advised that there is no exact match for the following feature combinations in her query:

make = Peugeot, origin = Japan
body style = hatchback, doors = 4

The user would be entitled to infer that a reasonable strategy might be to revise her query to:

{make = Peugeot, body style = hatchback, doors = 5}

Perhaps this would not be a trivial step for users with little experience of query formulation. However, the user is likely to gain additional clues to help her construct a more realistic query by browsing the list of most similar cases presented by the system in response to her original query (e.g., if one of them is a 5-door Peugeot hatchback). Also in contrast to approaches in which the absence of an exact match is regarded as a retrieval failure, the recommendation process is not interrupted in our approach by engaging the user in a recovery dialogue. Instead, the user has the options of choosing one of the recommended alternatives or revising her query in the light of the explanation provided.

Göker and Thompson (2000) make an important point about the environment in which the user interacts with the Adaptive Place Advisor and how this affects the way in which recommendations are delivered by the system. The fact that the user is engaged in an activity (driving her car) that must take priority over the recommendation process dictates that recommendations must be delivered in spoken dialogue rather than being presented visually. In this situation, the traditional approach of presenting the user with a ranked list of recommended alternatives, which may include none that exactly matches her requirements, loses much of its appeal. As envisaged in the Adaptive Place Advisor, the ability to take the initiative in dealing with retrieval failures, if allowed to occur, is likely to be an essential requirement in recommender systems that rely on spoken dialogue. Whether explanations of retrieval mismatches can effectively compensate for the limitations of

similarity-based retrieval in spoken dialogue remains an issue to be addressed by further research.

5 Conclusions

In the absence of a product that exactly matches the requirements of the user, existing recommender systems may not even acknowledge that a retrieval mismatch has occurred. Instead they often adopt a "*What you see is the best we've got*" approach that is unlikely to be considered helpful by users. Our approach to addressing this issue is based on explanations of retrieval mismatches that are designed to minimise cognitive load by providing the minimum information needed to avoid similar problems in further queries. In contrast to retrieval approaches that regard the absence of an exact match as a retrieval failure and engage the user in a recovery dialogue, our simple strategy of explaining *why* there is no exact match has the advantage of preserving the economy of dialogue associated with similarity-based retrieval.

Acknowledgements

The author would like to thank David W. Aha and Kalyan Moy Gupta for their insightful comments on a previous version of this paper. Thanks also to Lorraine McGinty for providing the PC case library used in our example recommendation dialogue.

References

1. Aha, D.W., Breslow, L.A., Muńoz-Avila, H.: Conversational Case-Based Reasoning. Applied Intelligence **14** (2001) 9-32
2. Bridge, D.: Towards Conversational Recommender Systems: a Dialogue Grammar Approach. In: Aha, D.W. (ed.) Proceedings of the EWCBR-02 Workshop on Mixed-Initiative Case-Based Reasoning (2002) 9-22
3. Bridge, D., Ferguson, A.: An Expressive Query Language for Product Recommender Systems. Artificial Intelligence Review, **18** (2002) 269-307
4. Burkhard, H.-D.: Extending Some Concepts of CBR - Foundations of Case Retrieval Nets. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (eds.) Case-Based Reasoning Technology. Springer-Verlag, Berlin Heidelberg New York (1998) 17-50
5. Göker, M.H., Thompson, C.A.: Personalized Conversational Case-Based Recommendation. In: Blanzieri, E., Portinale, L. (eds.) Advances in Case-Based Reasoning. LNAI, Vol. 1898. Springer-Verlag, Berlin Heidelberg (2000) 99-111
6. McGinty, L., Smyth, B.: Comparison-Based Recommendation. In: Craw, S., Preece, A. (eds.) Advances in Case-Based Reasoning. LNAI, Vol. 2416. Springer-Verlag, Berlin Heidelberg New York (2002) 575-58
7. McSherry, D.: Interactive Case-Based Reasoning in Sequential Diagnosis. Applied Intelligence **14** (2001) 65-76

8. McSherry, D.: Diversity-Conscious Retrieval. In: Craw, S., Preece, A. (eds.) Advances in Case-Based Reasoning. LNAI, Vol. 2416. Springer-Verlag, Berlin Heidelberg New York (2002a) 219-233
9. McSherry, D.: Recommendation Engineering. Proceedings of the Fifteenth European Conference on Artificial Intelligence. IOS Press (2002b) 86-90
10. McSherry, D.: Mixed-Initiative Dialogue in Case-Based Reasoning. In: Aha, D.W. (ed.) Proceedings of the EWCBR-02 Workshop on Mixed-Initiative Case-Based Reasoning (2002c) 1-8
11. McSherry, D.: Similarity and Compromise. In: Ashley, K., Bridge, D. (eds.) Case-Based Reasoning Research and Development. LNAI, Vol. 2689. Springer-Verlag, Berlin Heidelberg New York (2003)
12. Ricci, F., Arslan, B., Mirzadeh, N., Venturini, A.: ITR: A Case-Based Travel Advisory System. In: Craw, S., Preece, A. (eds.) Advances in Case-Based Reasoning. LNAI, Vol. 2416. Springer-Verlag, Berlin Heidelberg New York (2002) 613-627
13. Smyth, B., McClave, P.: Similarity vs. Diversity. In: Aha, D.W., Watson, I. (eds.) Case-Based Reasoning Research and Development. LNAI, Vol. 2080. Springer-Verlag, Berlin Heidelberg New York (2001) 347-361
14. Wilke, W., Lenz, M., Wess, S.: Intelligent Sales Support with CBR. In: Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., Wess, S. (eds.) Case-Based Reasoning Technology. Springer-Verlag, Berlin Heidelberg New York (1998) 91-113